ransactions on the Coinbase Cloud Developer Platform.  
ACCOUNT ABSTRACTION  
PAYMASTER  
Introduction to Providers  
🖊️ Brian Doyle  
Sep 12, 2024  
21 min read  
A tutorial that teaches what providers are, why you need one, and how to configure several providers and use them to connect to the blockchain.  
NODES  
FRONTEND  
Running a Base Node  
🖊️ taycaldwell & wbnns  
Sep 12, 2024  
7 min read  
A tutorial that teaches how to set up and run a Base Node.  
NODES  
Farcaster Frames: Building HyperFrames  
🖊️ Brian Doyle  
Aug 14, 2024  
16 min read  
A tutorial that teaches how to make cross-linked HyperFrames in an organized manner.  
FRAMES  
ONCHAINKIT  
Coinbase Smart Wallet  
🖊️ Brian Doyle  
Aug 14, 2024  
22 min read  
Learn to create an app that uses the Coinbase Smart Wallet and effectively manages assets and permissions for both native and new users of onchain apps  
NFT  
SMART WALLET  
PAYMASTER  
Use the Coinbase Smart Wallet and EOAs with OnchainKit  
🖊️ Brian Doyle  
Aug 14, 2024  
10 min read  
Learn how to use OnchainKit to easily handle the Coinbase Smart Wallet and EOA wallets including the Coinbase wallet, at the same time.  
ACCOUNT ABSTRACTION  
FRONTEND  
ONCHAINKIT  
SMART WALLET  
Coinbase Smart Wallet with RainbowKit  
🖊️ Brian Doyle  
Aug 14, 2024  
11 min read  
Learn how to configure RainbowKit to elegantly handle the Coinbase Smart Wallet and EOA wallets including the Coinbase wallet, at the same time.  
ACCOUNT ABSTRACTION  
FRONTEND  
SMART WALLET  
Farcaster Frames: Building an NFT airdrop Frame  
🖊️ Brian Doyle  
Aug 14, 2024  
23 min read  
A tutorial that teaches how to make a Farcaster Frame that allows you to mint and airdrop NFTs to users.  
FRAMES  
NFT  
ONCHAINKIT  
Farcaster Frames: Building a no-code minting Frame  
🖊️ Brian Doyle  
Aug 14, 2024  
6 min read  
A tutorial that teaches how to make a Farcaster Frame with an outbound link to an NFT minting website.  
FRAMES  
NFT  
ONCHAINKIT  
Farcaster Frames: Deploying to Vercel  
🖊️ Brian Doyle  
Aug 14, 2024  
9 min read  
A tutorial that teaches how to deploy a Farcaster Frame using Vercel.  
FRAMES  
ONCHAINKIT  
Farcaster Frames: Gating content and creating redirects  
🖊️ Brian Doyle  
Aug 14, 2024  
11 min read  
A tutorial that teaches how to create Frames with more advanced behaviors such as gating content based on a user's follows, likes, or recasts, and creating redirect buttons.  
FRAMES  
ONCHAINKIT  
Farcaster Frames: Making transactions  
🖊️ Brian Doyle  
Aug 14, 2024  
17 min read  
A tutorial that teaches how to invoke a wallet transaction from a Farcaster Frame.  
FRAMES  
ONCHAINKIT  
Farcaster Cast Actions: Create a Simple Cast Action  
🖊️ Brian Doyle  
Aug 14, 2024  
9 min read  
A tutorial that teaches how to make a simple Farcaster cast action.  
FRAMES  
ACTIONS  
ONCHAINKIT  
Verify a Smart Contract using Basescan API  
🖊️ hughescoin  
Sep 12, 2024  
12 min read  
A tutorial that teaches how to verify a smart contract using Basescan APIs.  
SMART CONTRACTS  
Simple Onchain NFTs  
🖊️ Brian Doyle  
Aug 14, 2024  
26 min read  
A tutorial that teaches how to make simple nfts that are procedurally generated and have onchain metadata and images.  
NFT  
Complex Onchain NFTs  
🖊️ Brian Doyle  
Aug 14, 2024  
39 min read  
A tutorial that teaches how to make complex nfts that are procedurally generated and have onchain metadata and images.  
NFT  
Signature Mint NFT  
🖊️ Brian Doyle  
Aug 14, 2024  
19 min read  
A tutorial that teaches how to create a signature mint, in which minters pay their own gas, but must first be given a valid signed authorization.  
NFT  
Building an onchain app using thirdweb  
🖊️ Taylor Caldwell  
Aug 14, 2024  
8 min read  
A tutorial that teaches how to build an NFT gallery app using thirdweb, including steps for creating an NFT collection, minting NFTs, and configuring the app for the Base testnet.  
SMART CONTRACTS  
NFT  
Thirdweb and Unreal - NFT Items  
🖊️ Brian Doyle  
Sep 12, 2024  
37 min read  
Learn how to use NFTs as in-game items using Thirdweb and Unreal.  
NFT  
SMART CONTRACTS  
Building dynamic NFTs  
🖊️ Luke Cassady-Dorion  
Sep 26, 2024  
14 min read  
A tutorial that teaches how to make dynamic NFTs that evolve based on onchain or offchain actions.  
NFT  
Create Email Marketing Campaigns Onchain using Coinbase Smart Wallet + Resend  
🖊️ hughescoin  
Sep 12, 2024  
21 min read  
A tutorial that teaches how to create a mailing list and email customers using Resend  
ACCOUNT ABSTRACTION  
Gate IRL Events with Nouns  
🖊️ Brian Doyle  
Aug 14, 2024  
19 min read  
Learn how to gate entry to an IRL event for members of a Nounish DAO.  
NFT  
SMART CONTRACTS  
FRONTEND  
How to Mint on Zora with an App  
🖊️ Brian Doyle  
Aug 13, 2024  
18 min read  
Learn to use Zora contracts inside your app to create secure, efficient, and feature-rich minting experiences for your users.  
NFT  
SMART WALLET  
Account Abstraction on Base using Biconomy  
🖊️ Taylor Caldwell  
Aug 14, 2024  
29 min read  
A tutorial that teaches how to implement Account Abstraction into a Base project using Biconomy paymasters, bundlers, and smart accounts.  
ACCOUNT ABSTRACTION  
Account Abstraction on Base using Particle Network  
🖊️ TABASCO  
Aug 14, 2024  
34 min read  
A walkthrough on Particle Network's Modular Smart Wallet-as-a-Service, leveraging account abstraction and social logins across various providers.  
ACCOUNT ABSTRACTION  
Account Abstraction on Base using Privy and the Base Paymaster  
🖊️ Brian Doyle and Aaron Hayslip  
Sep 12, 2024  
46 min read  
A tutorial that teaches how to implement Account Abstraction into a Base project using Privy and the Base paymaster.  
ACCOUNT ABSTRACTION  
Hardhat: Debugging smart contracts  
🖊️ Edson Alcala  
Jul 19, 2024  
15 min read  
A tutorial that teaches how to debug your smart contracts using Hardhat.  
SMART CONTRACTS  
Hardhat: Optimizing the gas usage of smart contracts  
🖊️ Edson Alcala and Brian Doyle  
Jul 11, 2024  
16 min read  
A tutorial that teaches how to optimize the gas usage of your smart contracts using Hardhat.  
SMART CONTRACTS  
Hardhat: Optimizing the size of smart contracts  
🖊️ Edson Alcala and Brian Doyle  
Aug 14, 2024  
22 min read  
A tutorial that teaches how to optimize the size of your smart contracts using Hardhat.  
SMART CONTRACTS  
Hardhat: Analyzing the test coverage of smart contracts  
🖊️ Edson Alcala  
Jul 19, 2024  
9 min read  
A tutorial that teaches how to profile the test coverage of your smart contracts using Hardhat and the Solidity Coverage plugin.  
SMART CONTRACTS  
Foundry: Setting up Foundry with Base  
🖊️ Edson Alcala  
Sep 26, 2024  
6 min read  
A tutorial that teaches how to set up your development environment to work with Foundry.  
SMART CONTRACTS  
Foundry: Testing smart contracts  
🖊️ Edson Alcala  
Jul 11, 2024  
9 min read  
A tutorial that teaches how to test your smart contracts using Foundry.  
SMART CONTRACTS  
Add Frames to A Basename  
🖊️ hughescoin  
Sep 26, 2024  
5 min read  
Learn how to customize your Basename by adding Frames, showcasing dynamic content with ease on Base.  
ACCOUNT ABSTRACTION  
PAYMASTER  
Sending messages and tokens from Base to other chains using Chainlink CCIP  
🖊️ Taylor Caldwell  
Aug 14, 2024  
32 min read  
A tutorial that teaches how to use Chainlink CCIP to perform cross-chain messaging and token transfers from Base Goerli testnet to Optimism Goerli testnet.  
CROSS-CHAIN  
Sending messages from Base to other chains using LayerZero V2  
🖊️ Taylor Caldwell  
Sep 26, 2024  
34 min read  
A tutorial that teaches how to use LayerZero V2 to perform cross-chain messaging from Base Goerli testnet to Optimism Goerli testnet.  
CROSS-CHAIN  
Deploy an Onchain App with Fleek  
🖊️ Brian Doyle  
Aug 14, 2024  
9 min read  
Learn how to deploy an onchain app using Fleek.  
FRONTEND  
Convert Farcaster Frame to Open Frame using OnchainKit  
🖊️ hughescoin  
Oct 15, 2024  
10 min read  
Learn how to convert your Farcaster Frame to an Open Frame using Onchain Kit, enabling broader compatibility and usage.  
FRONTEND  
Accessing real-world data using Chainlink Data Feeds  
🖊️ Taylor Caldwell  
Sep 26, 2024  
11 min read  
A tutorial that teaches how to use Chainlink Data Feeds to access real-world data, such as asset prices, directly from your smart contracts on the Base testnet.  
ORACLES  
Accessing real-time asset data using Pyth Price Feeds  
🖊️ Taylor Caldwell  
Aug 14, 2024  
13 min read  
A tutorial that teaches how to use Pyth Price Feeds to access real-time asset data, directly from your smart contracts on the Base testnet.  
ORACLES  
Generating random numbers contracts using Supra dVRF  
🖊️ Taylor Caldwell  
Aug 14, 2024  
19 min read  
A tutorial that teaches how to use Supra dVRF to serve random numbers using an onchain randomness generation mechanism directly within your smart contracts on the Base testnet.  
ORACLES  
VRF  
Deploy a Shopify Storefront with Coinbase Commerce  
🖊️ hughescoin  
Sep 12, 2024  
8 min read  
Learn how to launch a Shopify storefront that uses Coinbase Commerce as a crypto payment gateway.  
ECOMMERCE  
CRYPTO  
SHOPIFY  
ONCHAIN SUMMER  
OCS  
We use cookies and similar technologies on our websites to enhance and tailor your experience, analyze our traffic, and for security and marketing. You can choose not to allow some type of cookies by clicking Manage Settings. For more information see our Cookie Policy.  
Manage settings  
Accept all

URL: https://docs.base.org/tutorials/tags  
  
Ecosystem  
Bridge  
Builders  
About  
Socials  
Home  
Get Started  
Docs  
Learn  
Search  
K  
CONNECT  
Tags  
A  
account abstraction  
8  
actions  
1  
C  
cross-chain  
2  
crypto  
1  
E  
ecommerce  
1  
F  
frames  
7  
frontend  
7  
I  
identity  
1  
N  
nft  
11  
nodes  
2  
O  
ocs  
1  
onchain summer  
1  
OnchainKit  
8  
oracles  
3  
P  
Paymaster  
3  
S  
shopify  
1  
smart contracts  
15  
Smart Wallet  
4  
V  
vrf  
1  
We use cookies and similar technologies on our websites to enhance and tailor your experience, analyze our traffic, and for security and marketing. You can choose not to allow some type of cookies by clicking Manage Settings. For more information see our Cookie Policy.  
Manage settings  
Accept all

URL: https://docs.base.org/tutorials/tags/account-abstraction  
  
Ecosystem  
Bridge  
Builders  
About  
Socials  
Home  
Get Started  
Docs  
Learn  
Search  
K  
CONNECT  
8 docs tagged with "account abstraction"  
View All Tags  
Account Abstraction on Base using Biconomy  
A tutorial that teaches how to implement Account Abstraction into a Base project using Biconomy paymasters, bundlers, and smart accounts.  
Account Abstraction on Base using Particle Network  
A walkthrough on Particle Network's Modular Smart Wallet-as-a-Service, leveraging account abstraction and social logins across various providers.  
Account Abstraction on Base using Privy and the Base Paymaster  
A tutorial that teaches how to implement Account Abstraction into a Base project using Privy and the Base paymaster.  
Add Frames to A Basename  
Learn how to customize your Basename by adding Frames, showcasing dynamic content with ease on Base.  
Coinbase Smart Wallet with RainbowKit  
Learn how to configure RainbowKit to elegantly handle the Coinbase Smart Wallet and EOA wallets including the Coinbase wallet, at the same time.  
Create Email Marketing Campaigns Onchain using Coinbase Smart Wallet + Resend  
A tutorial that teaches how to create a mailing list and email customers using Resend  
Gasless Transactions on Base using a Paymaster  
Learn how to leverage the Base Paymaster for seamless, gasless transactions on the Coinbase Cloud Developer Platform.  
Use the Coinbase Smart Wallet and EOAs with OnchainKit  
Learn how to use OnchainKit to easily handle the Coinbase Smart Wallet and EOA wallets including the Coinbase wallet, at the same time.  
We use cookies and similar technologies on our websites to enhance and tailor your experience, analyze our traffic, and for security and marketing. You can choose not to allow some type of cookies by clicking Manage Settings. For more information see our Cookie Policy.  
Manage settings  
Accept all

URL: https://docs.base.org/tutorials/tags/actions  
  
Ecosystem  
Bridge  
Builders  
About  
Socials  
Home  
Get Started  
Docs  
Learn  
Search  
K  
CONNECT  
One doc tagged with "actions"  
View All Tags  
Farcaster Cast Actions: Create a Simple Cast Action  
A tutorial that teaches how to make a simple Farcaster cast action.  
We use cookies and similar technologies on our websites to enhance and tailor your experience, analyze our traffic, and for security and marketing. You can choose not to allow some type of cookies by clicking Manage Settings. For more information see our Cookie Policy.  
Manage settings  
Accept all

URL: https://docs.base.org/tutorials/tags/cross-chain  
  
Ecosystem  
Bridge  
Builders  
About  
Socials  
Home  
Get Started  
Docs  
Learn  
Search  
K  
CONNECT  
2 docs tagged with "cross-chain"  
View All Tags  
Sending messages and tokens from Base to other chains using Chainlink CCIP  
A tutorial that teaches how to use Chainlink CCIP to perform cross-chain messaging and token transfers from Base Goerli testnet to Optimism Goerli testnet.  
Sending messages from Base to other chains using LayerZero V2  
A tutorial that teaches how to use LayerZero V2 to perform cross-chain messaging from Base Goerli testnet to Optimism Goerli testnet.  
We use cookies and similar technologies on our websites to enhance and tailor your experience, analyze our traffic, and for security and marketing. You can choose not to allow some type of cookies by clicking Manage Settings. For more information see our Cookie Policy.  
Manage settings  
Accept all

URL: https://docs.base.org/tutorials/tags/crypto  
  
Ecosystem  
Bridge  
Builders  
About  
Socials  
Home  
Get Started  
Docs  
Learn  
Search  
K  
CONNECT  
One doc tagged with "crypto"  
View All Tags  
Deploy a Shopify Storefront with Coinbase Commerce  
Learn how to launch a Shopify storefront that uses Coinbase Commerce as a crypto payment gateway.  
We use cookies and similar technologies on our websites to enhance and tailor your experience, analyze our traffic, and for security and marketing. You can choose not to allow some type of cookies by clicking Manage Settings. For more information see our Cookie Policy.  
Manage settings  
Accept all

URL: https://docs.base.org/tutorials/tags/ecommerce  
  
Ecosystem  
Bridge  
Builders  
About  
Socials  
Home  
Get Started  
Docs  
Learn  
Search  
K  
CONNECT  
One doc tagged with "ecommerce"  
View All Tags  
Deploy a Shopify Storefront with Coinbase Commerce  
Learn how to launch a Shopify storefront that uses Coinbase Commerce as a crypto payment gateway.  
We use cookies and similar technologies on our websites to enhance and tailor your experience, analyze our traffic, and for security and marketing. You can choose not to allow some type of cookies by clicking Manage Settings. For more information see our Cookie Policy.  
Manage settings  
Accept all

URL: https://docs.base.org/tutorials/tags/frames  
  
Ecosystem  
Bridge  
Builders  
About  
Socials  
Home  
Get Started  
Docs  
Learn  
Search  
K  
CONNECT  
7 docs tagged with "frames"  
View All Tags  
Farcaster Cast Actions: Create a Simple Cast Action  
A tutorial that teaches how to make a simple Farcaster cast action.  
Farcaster Frames: Building a no-code minting Frame  
A tutorial that teaches how to make a Farcaster Frame with an outbound link to an NFT minting website.  
Farcaster Frames: Building an NFT airdrop Frame  
A tutorial that teaches how to make a Farcaster Frame that allows you to mint and airdrop NFTs to users.  
Farcaster Frames: Building HyperFrames  
A tutorial that teaches how to make cross-linked HyperFrames in an organized manner.  
Farcaster Frames: Deploying to Vercel  
A tutorial that teaches how to deploy a Farcaster Frame using Vercel.  
Farcaster Frames: Gating content and creating redirects  
A tutorial that teaches how to create Frames with more advanced behaviors such as gating content based on a user's follows, likes, or recasts, and creating redirect buttons.  
Farcaster Frames: Making transactions  
A tutorial that teaches how to invoke a wallet transaction from a Farcaster Frame.  
We use cookies and similar technologies on our websites to enhance and tailor your experience, analyze our traffic, and for security and marketing. You can choose not to allow some type of cookies by clicking Manage Settings. For more information see our Cookie Policy.  
Manage settings  
Accept all

URL: https://docs.base.org/terms-of-service  
  
Ecosystem  
Bridge  
Builders  
About  
Socials  
Home  
Get Started  
Docs  
Learn  
Search  
K  
CONNECT  
About Base  
Using Base  
Notices  
Preparing for fault proofs on Base Sepolia  
Building on Base  
Quickstart: Deploy on Base  
Network Information  
Base Contracts  
Fees  
Differences between Ethereum and Base  
Decentralizing Base with Optimism  
Tools  
Node Providers  
Block Explorers  
Network Faucets  
Oracles  
Data Indexers  
Cross-chain  
Account Abstraction  
Onramps  
User Onboarding  
Superchain Bridges  
Onchain Registry  
Basenames  
Toolchains  
Clients  
Tokens  
Bridging an L1 token to Base  
Adding tokens to Coinbase Wallet  
Contracts  
Security  
Status  
Brand Kit  
Terms of Service  
Privacy Policy  
Cookie Policy  
Sequencer, Testnet, Basenames Interface Terms  
Last Updated: August 5, 2024  
We’re excited you’re interested in Base, a layer-two optimistic rollup on the Ethereum public blockchain. While we do not control Base, these Terms of Service (“Terms”) constitute a legally binding contract made between you and Coinbase Technologies, Inc. (“Coinbase,” “we,” or “us”) that governs your access to and use of the Coinbase Sequencer, Base Testnet, and Basenames Interface, each of which is defined below (collectively, the “Services”). By using the Services in any way, you agree to be bound by these Terms. If you do not accept the terms and conditions of these Terms, you are not permitted to access or otherwise use the Services.  
BEFORE WE INCLUDE ANY OTHER DETAILS, WE WANT TO GIVE YOU NOTICE OF SOMETHING UP FRONT: BY AGREEING TO THESE TERMS, YOU AND WE AGREE TO RESOLVE ANY DISPUTES WE MAY HAVE WITH EACH OTHER VIA BINDING ARBITRATION OR IN SMALL CLAIMS COURT (INSTEAD OF A COURT OF GENERAL JURISDICTION), AND YOU AGREE TO DO SO AS AN INDIVIDUAL (INSTEAD OF, FOR EXAMPLE, AS A REPRESENTATIVE OR MEMBER OF A CLASS IN A CLASS ACTION). TO THE EXTENT THAT THE LAW ALLOWS, YOU ALSO WAIVE YOUR RIGHT TO A TRIAL BY JURY. FOR MORE INFORMATION, SEE OUR ARBITRATION AGREEMENT “DISPUTE RESOLUTION, ARBITRATION AGREEMENT, CLASS ACTION WAIVER, AND JURY TRIAL WAIVER.”  
1. Base and Bridging Smart Contracts  
The Base protocol (“Base”) is an open source, optimistic rollup protocol that operates with the Ethereum blockchain. The Base protocol includes protocol smart contracts that allow you to “bridge” (i.e., lock assets on one blockchain protocol and replicate them on another protocol) digital assets between Ethereum and/or Base (“Bridging Smart Contracts”). Neither Base nor the Bridging Smart Contracts are part of the Services. They are both operated through the use of certain open source software such as the OP Stack, an open sourced codebase approved by a decentralized, representative body of Optimism governance (the “Optimism Collective”), and a set of smart contracts that once deployed to the Base protocol are not controlled by Coinbase (even if Coinbase contributed to their initial development). Coinbase does not control what third parties may build on Base, the activity of such parties, any user transacting on Base, or any data stored on Base itself, and Coinbase does not take possession, custody, or control over any virtual currency or other digital asset on Base or the Bridging Smart Contracts, unless expressly stated in a written contract signed by Coinbase. You acknowledge and agree that Coinbase makes no representations or warranties with respect to Base or the Bridging Smart Contracts, and that, if you use Base or the Bridging Smart Contracts, you do so at your own risk.  
2. Basenames  
Basenames is an open source blockchain-based naming protocol that maintains a registry of all domains and subdomains on Base through a series of smart contracts deployed on Base. Basenames is not part of the Services. Users may, through interacting with the Basenames, search such registry, register domains and subdomains and manage their registered names. The Basenames interface located at https://base.org/names (the “Basenames Interface”) is one, but not the exclusive, means of accessing Basenames. You are responsible for conducting your own diligence on other interfaces enabling you to access Basenames to understand the fees and risks that they present. You understand that anyone can register and own a domain name (and its subdomains) that is not already registered on the registry maintained by Basenames. You further understand that names registered on the registry maintained by Basenames may expire and you are responsible for monitoring and renewing the registration of such names. You acknowledge that Coinbase is not able to forcibly remove, prevent or otherwise interfere with the ability of any person to register a domain name on the registry operated by Basenames and you hereby acknowledge that Coinbase will not be liable for any claims or damages whatsoever associated with your use, inability to use any domain names subject of registration, or to be registered, on the registry maintained by Basenames. You agree that Basenames is purely non-custodial, meaning you are solely responsible for the custody of the cryptographic private keys to the digital asset wallets you hold and use to access Basenames.  
3. Who May Use the Services  
You may only use the Services if you are legally capable of forming a binding contract with Coinbase in your respective jurisdiction which may require your parents consent if you’re not the legal age of majority (which in many jurisdictions is 18), and not barred from using the Services under the laws of any applicable jurisdiction, for example, that you do not appear on the U.S. Treasury Department’s list of Specially Designated Nationals and are not located or organized in a U.S. sanctioned jurisdiction. If you are using the Services on behalf of an entity or other organization, you agree to these Terms for that entity or organization and represent to Coinbase that you have the authority to bind that entity or organization to these Terms.  
4. Rights We Grant You  
As between you and us, Coinbase is the owner of the Services, including all related intellectual property rights and proprietary content, information, material, software, images, text, graphics, illustrations, logos, trademarks (including the Base logo, the Base name, the Coinbase logo, the Coinbase name, and any other Coinbase or Base marks), service marks, copyrights, photographs, audio, video, music, and the “look and feel” of the Services. We hereby permit you to use and access the Services, provided that you comply with these Terms. If any software, content or other materials owned or controlled by us are distributed to you as part of your use of the Services, we hereby grant you a non-sublicensable, non-transferable, and non-exclusive right and license to execute, access and display such software, content and materials provided to you as part of the Services, in each case for the sole purpose of enabling you to use the Services as permitted by these Terms. To use any parts of the contents of the Services other than for personal and non-commercial use, you must seek permission from Coinbase in writing. Coinbase reserves the right to refuse permission without providing any reasons.  
5. Accessing the Services  
To access the Services, Base, or the Bridging Smart Contracts you must connect a compatible cryptocurrency wallet software (“Wallet”). Your relationship with any given Wallet provider is governed by the applicable terms of that Wallet provider, not these Terms. You are responsible for maintaining the confidentiality of any private key controlled by your Wallet and are fully responsible for any and all messages or conduct signed with your private key. We accept no responsibility or liability to you in connection with your use of a Wallet, and make no representations and warranties regarding how the Services, Base, or the Bridging Smart Contracts will operate or be compatible with any specific Wallet. We reserve the right, in our sole discretion, to prohibit certain Wallet addresses from being able to use or engage in transactions via the Coinbase Sequencer or from using other aspects of the Services.  
As between you and Coinbase, you retain ownership and all intellectual property rights to the content and materials you submit to the Services. But, you grant us a limited, non-exclusive, worldwide, royalty free license to use your content solely for the purpose of operating the Services (i.e., the Sequencer and Base Testnet) for so long as we operate the Services. To avoid any doubt, this license does not allow us to use your intellectual property beyond operating the Services (e.g., in advertisements).  
6. The Services  
Coinbase offers the following Services that enable you to access and interact with Base and the Bridging Smart Contracts:  
The Sequencer: The Coinbase Sequencer is a node operated by Coinbase that receives, records, and reports transactions on Base. While The Coinbase Sequencer is, initially, the only sequencer node supporting transactions on Base, additional nodes may be provided by third parties in the future and there are other mechanisms for submitting transactions through Ethereum. The Coinbase Sequencer does not store, take custody of, control, send, or receive your virtual currency, except for receiving applicable gas fees. It also does not have the ability to modify, reverse, or otherwise alter any submitted transactions, and will not have access to your private key or the ability to control value on your behalf. We reserve the right to charge and modify the fees in connection with your use of the Coinbase Sequencer. These fees may also be subject to taxes under applicable law.  
Base Testnet: The Base Testnet is a test environment that allows you to build applications integrated with Base. You are permitted to access and use the Base Testnet only to test and improve the experience, security, and design of Base or applications built on Base, subject to these Terms. Base Testnet Tokens will not be converted into any future rewards offered by Coinbase. Coinbase may change, discontinue, or terminate, temporarily or permanently, all or any part of the Base Testnet, at any time and without notice.  
Basenames Interface: The Basenames Interface is a web application and graphical user display operated by Coinbase and located at base.org/names. It enables you to interact with Basenames by creating blockchain messages that you can sign and broadcast to Base using your Wallet. The Basenames Interface will not have access to your private key at any point.  
7. Acceptable Use  
You agree that you will not use the Services in any manner or for any purpose other than as expressly permitted by these Terms. That means, among other things, you will not use the Services to do or encourage any of the following:  
Infringe or violate the intellectual property rights or any other rights of anyone else (including Coinbase) or attempt to decompile, disassemble, or reverse engineer the Services;  
Violate any applicable law or regulation, including without limitation, any applicable anti-money laundering laws, anti-terrorism laws, export control laws, end user restrictions, privacy laws or economic sanctions laws/regulations, including those administered by the U.S. Department of Treasury’s Office of Foreign Assets Control;  
Use the Services in a way that is dangerous, harmful, fraudulent, misleading, deceptive, threatening, harassing, defamatory, obscene, or otherwise objectionable;  
Violate, compromise, or interfere with the security, integrity, or availability of any computer, network, or technology associated with the Services, including using the Services in a manner that constitutes excessive or abusive usage, attempts to disrupt, attack, or interfere with other users, or otherwise impacts the stability of the Services.  
Use any Coinbase brands, logos, or trademarks (or any brands, logos, or trademarks that are confusingly similar) without our express prior written approval, which we may withhold at our discretion for any reason.  
8. Release and Assumption of Risk  
‍By using the Services, Base, or the Bridging Smart Contracts, you represent that you understand there are risks inherent in using cryptographic and public blockchain-based systems, including, but not limited, to the Services and digital assets such as bitcoin (BTC) and ether (ETH). You expressly agree that you assume all risks in connection with your access and use of Base, the Bridging Smart Contracts, Basenames, and the separate Services offered by Coinbase. That means, among other things, you understand and acknowledge that:  
The Base, the Bridging Smart Contracts, Basenames, and the separate Services may be subject to cyberattacks and exploits, which could result in the irrevocable loss or reduction in value of your digital assets or in additional copies of your digital assets being created or bridged without your consent.  
Base is subject to periodic upgrades by the Optimism Collective. The Optimism Collective may approve a protocol upgrade that, if implemented, may significantly impacts Base, and may introduce other risks, bugs, malfunctions, cyberattack vectors, or other changes to Base that could disrupt the operation of Base, the Bridging Smart Contracts, Basenames, or the Services or otherwise cause you damage or loss.  
If you lose your Wallet seed phrase, private keys, or password, you might permanently be unable to access your digital assets. You bear sole responsibility for safeguarding and ensuring the security of your Wallet.  
You further expressly waive and release Coinbase, its parents, affiliates, related companies, their officers, directors, members, employees, consultants, representatives. agents, partners, licensors, and each of their respective successors and assigns (collectively, the “Coinbase Entities”) from any and all liability, claims, causes of action, or damages arising from or in any way related to your use of the Services, and your interaction with Base, the Bridging Smart Contracts, or Basenames. Also, to the extent applicable, you shall and hereby do waive the benefits and protections of California Civil Code § 1542, which provides: “[a] general release does not extend to claims that the creditor or releasing party does not know or suspect to exist in his or her favor at the time of executing the release and that, if known by him or her, would have materially affected his or her settlement with the debtor or released party.”  
9. Interactions with Other Users  
You are responsible for your interactions with other users on or through the Services. While we reserve the right to monitor interactions between users, we are not obligated to do so, and we cannot be held liable for your interactions with other users, or for any user’s actions or inactions. If you have a dispute with one or more users, you release us (and our affiliates and subsidiaries, and our and their respective officers, directors, employees and agents) from claims, demands and damages (actual and consequential) of every kind and nature, known and unknown, arising out of or in any way connected with such disputes. In entering into this release you expressly waive any protections (whether statutory or otherwise) that would otherwise limit the coverage of this release to include only those claims which you may know or suspect to exist in your favor at the time of agreeing to this release.  
10. Feedback  
Any questions, comments, suggestions, ideas, feedback, reviews, or other information about the Services, provided by you to Coinbase, are non-confidential and Coinbase will be entitled to the unrestricted use and dissemination of these submissions for any purpose, commercial or otherwise, without acknowledgment, attribution, or compensation to you.  
11. Privacy  
For more information regarding our collection, use, and disclosure of personal data and certain other data, please see our Privacy Policy. The processing of personal data by Coinbase as a processor will be subject to any data processing agreement that you enter into with Coinbase.  
12. Third-Party Services  
The Services may provide access to services, sites, technology, applications and resources that are provided or otherwise made available by third parties (“Third-Party Services”). Your access and use of Third-Party Services may also be subject to additional terms and conditions, privacy policies, or other agreements with such third parties. Coinbase has no control over and is not responsible for such Third-Party Services, including for the accuracy, availability, reliability, or completeness of information or content shared by or available through Third-Party Services, or on the privacy practices of Third-Party Services. We encourage you to review the privacy policies of Third-Party Services prior to using such services. You, and not Coinbase, will be responsible for any and all costs and charges associated with your use of any Third-Party Services. The integration or inclusion of such Third-Party Services does not imply an endorsement or recommendation. Any dealings you have with third parties while using the Services — including if a Third-Party Service may have infringed your intellectual property rights — are between you and the third party. Coinbase will not be responsible or liable, directly or indirectly, for any damage or loss caused or alleged to be caused by or in connection with use of or reliance on any Third-Party Services.  
13. Additional Services  
We or our affiliates may offer additional services that interact with Base, which may require you to agree to additional terms. If, while using an additional service, there is a conflict between these Terms and the additional terms covering that service, the additional terms will prevail.  
14. Indemnification  
To the fullest extent permitted by applicable laws, you will indemnify and hold the Coinbase Entities harmless from and against any claims, disputes, demands, liabilities, damages, losses, and costs and expenses, including, without limitation, reasonable legal and accounting fees arising out of or in any way connected with (a) your access to or use of the Services, (b) your violation of these Terms, or (c) your negligence or willful misconduct. If you are obligated to indemnify any Coinbase Entity hereunder, then you agree that Coinbase (or, at its discretion, the applicable Coinbase Entity) will have the right, in its sole discretion, to control any action or proceeding and to determine whether Coinbase wishes to settle, and if so, on what terms, and you agree to fully cooperate with Coinbase in the defense or settlement of such claim.  
15. Warranty Disclaimers  
TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, BASE, THE BRIDGING SMART CONTRACTS, BASENAMES, AND THE SERVICES ARE PROVIDED ON AN “AS IS” AND “AS AVAILABLE” BASIS WITHOUT ANY REPRESENTATION OR WARRANTY, WHETHER EXPRESS, IMPLIED OR STATUTORY. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, COINBASE SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF TITLE, MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND/OR NON-INFRINGEMENT. THE COINBASE ENTITIES DO NOT MAKE ANY REPRESENTATIONS OR WARRANTIES THAT (I) ACCESS TO THE SERVICES, BASE, THE BRIDGING SMART CONTRACTS, OR BASENAMES WILL BE CONTINUOUS, UNINTERRUPTED, OR TIMELY; (II) THE SERVICES, BASE, THE BRIDGING SMART CONTRACTS, OR BASENAMES WILL BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM OR OTHER SERVICES, INCLUDING ANY WALLETS; (III) THE SERVICES, BASE, THE BRIDGING SMART CONTRACTS, OR BASENAMES WILL BE SECURE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE; (IV) THE SERVICES, BASE, THE BRIDGING SMART CONTRACTS, OR BASENAMES WILL PREVENT ANY UNAUTHORIZED ACCESS TO, ALTERATION OF, OR THE DELETION, DESTRUCTION, DAMAGE, LOSS OR FAILURE TO STORE ANY OF YOUR CONTENT OR OTHER DATA; OR (V) THAT THE SERVICES, BASE, THE BRIDGING SMART CONTRACTS, OR BASENAMES WILL PROTECT YOUR ASSETS FROM THEFT, HACKING, CYBER ATTACK, OR OTHER FORM OF LOSS OR DEVALUATION CAUSED BY THIRD-PARTY CONDUCT.  
16. Limitation of Liability  
TO THE MAXIMUM EXTENT PERMITTED BY LAW, NEITHER THE COINBASE ENTITIES NOR ITS SERVICE PROVIDERS INVOLVED IN CREATING, PRODUCING, OR DELIVERING THE SERVICES WILL BE LIABLE FOR ANY INCIDENTAL, SPECIAL, EXEMPLARY OR CONSEQUENTIAL DAMAGES, OR DAMAGES FOR LOST PROFITS, LOST REVENUES, LOST SAVINGS, LOST BUSINESS OPPORTUNITY, LOSS OF DATA OR GOODWILL, SERVICE INTERRUPTION, COMPUTER DAMAGE OR SYSTEM FAILURE, INTELLECTUAL PROPERTY INFRINGEMENT, OR THE COST OF SUBSTITUTE SERVICES OF ANY KIND ARISING OUT OF OR IN CONNECTION WITH THESE TERMS OR FROM THE USE OF OR INABILITY TO USE THE SERVICES, BASE, THE BRIDGING SMART CONTRACTS, OR BASENAMES, WHETHER BASED ON WARRANTY, CONTRACT, TORT (INCLUDING NEGLIGENCE), PRODUCT LIABILITY OR ANY OTHER LEGAL THEORY, AND WHETHER OR NOT THE COINBASE ENTITIES OR ITS SERVICE PROVIDERS HAVE BEEN INFORMED OF THE POSSIBILITY OF SUCH DAMAGE, EVEN IF A LIMITED REMEDY SET FORTH HEREIN IS FOUND TO HAVE FAILED OF ITS ESSENTIAL PURPOSE.  
TO THE MAXIMUM EXTENT PERMITTED BY LAW, IN NO EVENT WILL THE COINBASE ENTITIES’ TOTAL LIABILITY ARISING OUT OF OR IN CONNECTION WITH THESE TERMS OR FROM THE USE OF OR INABILITY TO USE THE SERVICES, BASE, THE BRIDGING SMART CONTRACTS, OR BASENAMES EXCEED THE AMOUNTS YOU HAVE PAID OR ARE PAYABLE BY YOU TO THE COINBASE ENTITIES FOR USE OF THE SERVICES OR ONE HUNDRED DOLLARS ($100), WHICHEVER IS HIGHER.  
THE EXCLUSIONS AND LIMITATIONS OF DAMAGES SET FORTH ABOVE ARE FUNDAMENTAL ELEMENTS OF THE BASIS OF THE BARGAIN BETWEEN COINBASE AND YOU.  
IF ANY PORTION OF THESE SECTIONS IS HELD TO BE INVALID UNDER THE LAWS OF YOUR STATE OF RESIDENCE, THE INVALIDITY OF SUCH PORTION WILL NOT AFFECT THE VALIDITY OF THE REMAINING PORTIONS OF THE APPLICABLE SECTIONS. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL OR CERTAIN OTHER DAMAGES, SO THE ABOVE LIMITATIONS AND EXCLUSIONS MAY NOT APPLY TO YOU.  
17. Changes to Terms  
We reserve the right, in our sole discretion, to change these Terms at any time and your continued use of the Services after the date any such changes become effective constitutes your acceptance of the new Terms. You should periodically visit this page to review the current Terms so you are aware of any revisions. If you do not agree to abide by these or any future Terms, you are not permitted to access, browse, or use (or continue to access, browse, or use) the Services.  
18. Notice  
Any notices or other communications provided by us under these Terms, including those regarding modifications to these Terms, will be posted online, in the Services, or through other electronic communication. You agree and consent to receive electronically all communications, agreements, documents, notices and disclosures that we provide in connection with your use of the Services.  
19. Entire Agreement.  
These Terms and any other documents incorporated by reference comprise the entire understanding and agreement between you and Coinbase as to the subject matter hereof, and supersedes any and all prior discussions, agreements and understandings of any kind (including without limitation any prior versions of these Terms), between you and Coinbase. Section headings in these Terms are for convenience only and shall not govern the meaning or interpretation of any provision of these Terms.  
20. Assignment  
We reserve the right to assign our rights without restriction, including without limitation to any Coinbase affiliates or subsidiaries, or to any successor in interest of any business associated with the Services. In the event that Coinbase is acquired by or merged with a third party entity, we reserve the right, in any of these circumstances, to transfer or assign the information we have collected from you as part of such merger, acquisition, sale, or other change of control. You may not assign any rights and/or licenses granted under these Terms. Any attempted transfer or assignment by you in violation hereof shall be null and void. Subject to the foregoing, these Terms will bind and inure to the benefit of the parties, their successors and permitted assigns.  
21. Severability  
If any provision of these Terms is determined to be invalid or unenforceable under any rule, law, or regulation of any local, state, or federal government agency, such provision will be changed and interpreted to accomplish the objectives of the provision to the greatest extent possible under any applicable law and the validity or enforceability of any other provision of these Terms shall not be affected.  
22. Termination; Survival  
We may suspend or terminate your access to and use of the Services at our sole discretion, at any time and without notice to you. Upon any termination, discontinuation or cancellation of the Services, Sections 7 through 27 of the Terms will survive.  
23. Governing Law  
You agree that the laws of the State of California, without regard to principles of conflict of laws, will govern these Terms and any Dispute, except to the extent governed by federal law.  
24. Force Majeure  
We shall not be liable for delays, failure in performance or interruption of service which result directly or indirectly from any cause or condition beyond our reasonable control, including but not limited to, significant market volatility, act of God, act of civil or military authorities, act of terrorists, civil disturbance, war, strike or other labor dispute, fire, interruption in telecommunications or Internet services or network provider services, failure of equipment and/or software, pandemic, other catastrophe or any other occurrence which is beyond our reasonable control and shall not affect the validity and enforceability of any remaining provisions.  
25. Non-Waiver of Rights  
These Terms shall not be construed to waive rights that cannot be waived under applicable laws, including applicable state money transmission laws in the state where you are located. In addition, our failure to insist upon or enforce strict performance by you of any provision of these Terms or to exercise any right under these Terms will not be construed as a waiver or relinquishment to any extent of our right to assert or rely upon any such provision or right in that or any other instance.  
26. Relationship of the Parties  
Coinbase is an independent contractor for all purposes. Nothing in these Terms is intended to or shall operate to create a partnership or joint venture between you and Coinbase, or authorize you to act as agent of Coinbase. These Terms are not intended to, and do not, create or impose any fiduciary duties on us. To the fullest extent permitted by law, you acknowledge and agree that we owe no fiduciary duties or liabilities to you or any other party, and that to the extent any such duties or liabilities may exist at law or in equity, those duties and liabilities are hereby irrevocably disclaimed, waived, and foregone. You further agree that the only duties and obligations that we owe you are those set out expressly in these Terms.  
26. Dispute Resolution, Arbitration Agreement, Class Action Waiver, And Jury Trial Waiver  
If you have a dispute with us, you agree to first contact Coinbase Support via our Customer Support page (https://help.coinbase.com). If Coinbase Support is unable to resolve your dispute, you agree to follow our Formal Complaint Process. You begin this process by submitting our complaint form. If you would prefer to send a written complaint via mail, please include as much information as possible in describing your complaint, including your support ticket number, how you would like us to resolve the complaint, and any other relevant information to us at 82 Nassau St #61234, New York, NY 10038. The Formal Complaint Process is completed when Coinbase responds to your complaint or 45 business days after the date we receive your complaint, whichever occurs first. You agree to complete the Formal Complaint Process before filing an arbitration demand or action in small claims court.  
Disputes with Users Who Reside in the United States or Canada  
If you reside in the United States or Canada, and if you have a dispute with us or if we have a dispute with you, the dispute shall be resolved through binding arbitration or in small claims court pursuant to the Arbitration Agreement in Appendix 1 below.  
As an illustration only, the following is a summary of some of the terms of the Arbitration Agreement:  
Disputes will be resolved individually (in other words, you are waiving your right to proceed against Coinbase in a class action). However, if you or we bring a coordinated group of arbitration demands with other claimants, you and we agree that the American Arbitration Association (AAA) must batch your or our arbitration demand with up to 100 other claimants to increase the efficiency and resolution of such claims.  
Certain disputes must be decided before a court, including (1) any claim that the class action waiver is unenforceable, (2) any dispute about the payment of arbitration fees, (3) any dispute about whether you have completed the prerequisites to arbitration (such as exhausting the support and Formal Complaint processes), (4) any dispute about which version of the Arbitration Agreement applies, and (5) any dispute about whether a dispute is subject to the Arbitration Agreement in the first instance.  
In the event that a dispute is filed with a court that does not fall into one of the above five categories, either you or Coinbase may move to compel the court to order arbitration. If the court issues an order compelling arbitration, the prevailing party on the motion to compel may recover its reasonable attorneys’ fees and costs.  
Disputes with Users Who Reside Outside the United States and Canada  
If you do not reside in the United States or Canada, the Arbitration Agreement in Appendix 1 does not apply to you and you may resolve any claim you have with us relating to, arising out of, or in any way in connection with our Terms, us, or our Services in a court of competent jurisdiction.  
We use cookies and similar technologies on our websites to enhance and tailor your experience, analyze our traffic, and for security and marketing. You can choose not to allow some type of cookies by clicking Manage Settings. For more information see our Cookie Policy.  
Manage settings  
Accept all

URL: https://docs.base.org/tutorials/tags/frontend  
  
Ecosystem  
Bridge  
Builders  
About  
Socials  
Home  
Get Started  
Docs  
Learn  
Search  
K  
CONNECT  
7 docs tagged with "frontend"  
View All Tags  
Coinbase Smart Wallet with RainbowKit  
Learn how to configure RainbowKit to elegantly handle the Coinbase Smart Wallet and EOA wallets including the Coinbase wallet, at the same time.  
Convert Farcaster Frame to Open Frame using OnchainKit  
Learn how to convert your Farcaster Frame to an Open Frame using Onchain Kit, enabling broader compatibility and usage.  
Create a Basename Profile Component  
Learn how to create a component that displays social media and profile information for a given basename.  
Deploy an Onchain App with Fleek  
Learn how to deploy an onchain app using Fleek.  
Gate IRL Events with Nouns  
Learn how to gate entry to an IRL event for members of a Nounish DAO.  
Introduction to Providers  
A tutorial that teaches what providers are, why you need one, and how to configure several providers and use them to connect to the blockchain.  
Use the Coinbase Smart Wallet and EOAs with OnchainKit  
Learn how to use OnchainKit to easily handle the Coinbase Smart Wallet and EOA wallets including the Coinbase wallet, at the same time.  
We use cookies and similar technologies on our websites to enhance and tailor your experience, analyze our traffic, and for security and marketing. You can choose not to allow some type of cookies by clicking Manage Settings. For more information see our Cookie Policy.  
Manage settings  
Accept all

URL: https://docs.base.org/tutorials/tags/nft  
  
Ecosystem  
Bridge  
Builders  
About  
Socials  
Home  
Get Started  
Docs  
Learn  
Search  
K  
CONNECT  
11 docs tagged with "nft"  
View All Tags  
Building an onchain app using thirdweb  
A tutorial that teaches how to build an NFT gallery app using thirdweb, including steps for creating an NFT collection, minting NFTs, and configuring the app for the Base testnet.  
Building dynamic NFTs  
A tutorial that teaches how to make dynamic NFTs that evolve based on onchain or offchain actions.  
Coinbase Smart Wallet  
Learn to create an app that uses the Coinbase Smart Wallet and effectively manages assets and permissions for both native and new users of onchain apps  
Complex Onchain NFTs  
A tutorial that teaches how to make complex nfts that are procedurally generated and have onchain metadata and images.  
Farcaster Frames: Building a no-code minting Frame  
A tutorial that teaches how to make a Farcaster Frame with an outbound link to an NFT minting website.  
Farcaster Frames: Building an NFT airdrop Frame  
A tutorial that teaches how to make a Farcaster Frame that allows you to mint and airdrop NFTs to users.  
Gate IRL Events with Nouns  
Learn how to gate entry to an IRL event for members of a Nounish DAO.  
How to Mint on Zora with an App  
Learn to use Zora contracts inside your app to create secure, efficient, and feature-rich minting experiences for your users.  
Signature Mint NFT  
A tutorial that teaches how to create a signature mint, in which minters pay their own gas, but must first be given a valid signed authorization.  
Simple Onchain NFTs  
A tutorial that teaches how to make simple nfts that are procedurally generated and have onchain metadata and images.  
Thirdweb and Unreal - NFT Items  
Learn how to use NFTs as in-game items using Thirdweb and Unreal.  
We use cookies and similar technologies on our websites to enhance and tailor your experience, analyze our traffic, and for security and marketing. You can choose not to allow some type of cookies by clicking Manage Settings. For more information see our Cookie Policy.  
Manage settings  
Accept all

URL: https://docs.base.org/tutorials/tags/nodes  
  
Ecosystem  
Bridge  
Builders  
About  
Socials  
Home  
Get Started  
Docs  
Learn  
Search  
K  
CONNECT  
2 docs tagged with "nodes"  
View All Tags  
Introduction to Providers  
A tutorial that teaches what providers are, why you need one, and how to configure several providers and use them to connect to the blockchain.  
Running a Base Node  
A tutorial that teaches how to set up and run a Base Node.  
We use cookies and similar technologies on our websites to enhance and tailor your experience, analyze our traffic, and for security and marketing. You can choose not to allow some type of cookies by clicking Manage Settings. For more information see our Cookie Policy.  
Manage settings  
Accept all

URL: https://docs.base.org/tutorials/tags/ocs  
  
Ecosystem  
Bridge  
Builders  
About  
Socials  
Home  
Get Started  
Docs  
Learn  
Search  
K  
CONNECT  
One doc tagged with "ocs"  
View All Tags  
Deploy a Shopify Storefront with Coinbase Commerce  
Learn how to launch a Shopify storefront that uses Coinbase Commerce as a crypto payment gateway.  
We use cookies and similar technologies on our websites to enhance and tailor your experience, analyze our traffic, and for security and marketing. You can choose not to allow some type of cookies by clicking Manage Settings. For more information see our Cookie Policy.  
Manage settings  
Accept all

URL: https://docs.base.org/tutorials/tags/onchain-kit  
  
Ecosystem  
Bridge  
Builders  
About  
Socials  
Home  
Get Started  
Docs  
Learn  
Search  
K  
CONNECT  
8 docs tagged with "OnchainKit"  
View All Tags  
Farcaster Cast Actions: Create a Simple Cast Action  
A tutorial that teaches how to make a simple Farcaster cast action.  
Farcaster Frames: Building a no-code minting Frame  
A tutorial that teaches how to make a Farcaster Frame with an outbound link to an NFT minting website.  
Farcaster Frames: Building an NFT airdrop Frame  
A tutorial that teaches how to make a Farcaster Frame that allows you to mint and airdrop NFTs to users.  
Farcaster Frames: Building HyperFrames  
A tutorial that teaches how to make cross-linked HyperFrames in an organized manner.  
Farcaster Frames: Deploying to Vercel  
A tutorial that teaches how to deploy a Farcaster Frame using Vercel.  
Farcaster Frames: Gating content and creating redirects  
A tutorial that teaches how to create Frames with more advanced behaviors such as gating content based on a user's follows, likes, or recasts, and creating redirect buttons.  
Farcaster Frames: Making transactions  
A tutorial that teaches how to invoke a wallet transaction from a Farcaster Frame.  
Use the Coinbase Smart Wallet and EOAs with OnchainKit  
Learn how to use OnchainKit to easily handle the Coinbase Smart Wallet and EOA wallets including the Coinbase wallet, at the same time.  
We use cookies and similar technologies on our websites to enhance and tailor your experience, analyze our traffic, and for security and marketing. You can choose not to allow some type of cookies by clicking Manage Settings. For more information see our Cookie Policy.  
Manage settings  
Accept all

URL: https://docs.base.org/tutorials/tags/onchain-summer  
  
Ecosystem  
Bridge  
Builders  
About  
Socials  
Home  
Get Started  
Docs  
Learn  
Search  
K  
CONNECT  
One doc tagged with "onchain summer"  
View All Tags  
Deploy a Shopify Storefront with Coinbase Commerce  
Learn how to launch a Shopify storefront that uses Coinbase Commerce as a crypto payment gateway.  
We use cookies and similar technologies on our websites to enhance and tailor your experience, analyze our traffic, and for security and marketing. You can choose not to allow some type of cookies by clicking Manage Settings. For more information see our Cookie Policy.  
Manage settings  
Accept all

URL: https://docs.base.org/tutorials/tags/oracles  
  
Ecosystem  
Bridge  
Builders  
About  
Socials  
Home  
Get Started  
Docs  
Learn  
Search  
K  
CONNECT  
3 docs tagged with "oracles"  
View All Tags  
Accessing real-time asset data using Pyth Price Feeds  
A tutorial that teaches how to use Pyth Price Feeds to access real-time asset data, directly from your smart contracts on the Base testnet.  
Accessing real-world data using Chainlink Data Feeds  
A tutorial that teaches how to use Chainlink Data Feeds to access real-world data, such as asset prices, directly from your smart contracts on the Base testnet.  
Generating random numbers contracts using Supra dVRF  
A tutorial that teaches how to use Supra dVRF to serve random numbers using an onchain randomness generation mechanism directly within your smart contracts on the Base testnet.  
We use cookies and similar technologies on our websites to enhance and tailor your experience, analyze our traffic, and for security and marketing. You can choose not to allow some type of cookies by clicking Manage Settings. For more information see our Cookie Policy.  
Manage settings  
Accept all

URL: https://docs.base.org/tutorials/tags/paymaster  
  
Ecosystem  
Bridge  
Builders  
About  
Socials  
Home  
Get Started  
Docs  
Learn  
Search  
K  
CONNECT  
3 docs tagged with "Paymaster"  
View All Tags  
Add Frames to A Basename  
Learn how to customize your Basename by adding Frames, showcasing dynamic content with ease on Base.  
Coinbase Smart Wallet  
Learn to create an app that uses the Coinbase Smart Wallet and effectively manages assets and permissions for both native and new users of onchain apps  
Gasless Transactions on Base using a Paymaster  
Learn how to leverage the Base Paymaster for seamless, gasless transactions on the Coinbase Cloud Developer Platform.  
We use cookies and similar technologies on our websites to enhance and tailor your experience, analyze our traffic, and for security and marketing. You can choose not to allow some type of cookies by clicking Manage Settings. For more information see our Cookie Policy.  
Manage settings  
Accept all

URL: https://docs.base.org/tutorials/tags/shopify  
  
Ecosystem  
Bridge  
Builders  
About  
Socials  
Home  
Get Started  
Docs  
Learn  
Search  
K  
CONNECT  
One doc tagged with "shopify"  
View All Tags  
Deploy a Shopify Storefront with Coinbase Commerce  
Learn how to launch a Shopify storefront that uses Coinbase Commerce as a crypto payment gateway.  
We use cookies and similar technologies on our websites to enhance and tailor your experience, analyze our traffic, and for security and marketing. You can choose not to allow some type of cookies by clicking Manage Settings. For more information see our Cookie Policy.  
Manage settings  
Accept all

URL: https://docs.base.org/tutorials/tags/smart-contracts  
  
Ecosystem  
Bridge  
Builders  
About  
Socials  
Home  
Get Started  
Docs  
Learn  
Search  
K  
CONNECT  
15 docs tagged with "smart contracts"  
View All Tags  
Building an onchain app using thirdweb  
A tutorial that teaches how to build an NFT gallery app using thirdweb, including steps for creating an NFT collection, minting NFTs, and configuring the app for the Base testnet.  
Deploying a smart contract using Foundry  
A tutorial that teaches how to deploy a smart contract on the Base test network using Foundry. Includes instructions for setting up the environment, compiling, and deploying the smart contract.  
Deploying a smart contract using Hardhat  
A tutorial that teaches how to deploy a smart contract on the Base test network using Hardhat. Includes instructions for setting up the environment, compiling, and deploying the smart contract.  
Deploying a smart contract using Remix  
A tutorial that teaches how to deploy a smart contract on the Base test network using Remix IDE. Includes instructions for setting up the environment, compiling, and deploying the smart contract.  
Deploying a smart contract using Tenderly  
A tutorial that teaches how to deploy smart contracts using Tenderly DevNets. This page covers setup, debugging, transaction simulations, and continuous integration for smart contract development on Base Network.  
Deploying a smart contract using thirdweb  
A tutorial that teaches how to deploy and interact with smart contracts using the thirdweb CLI and SDK. Includes instructions for project creation, contract deployment on the Base test network.  
Foundry: Setting up Foundry with Base  
A tutorial that teaches how to set up your development environment to work with Foundry.  
Foundry: Testing smart contracts  
A tutorial that teaches how to test your smart contracts using Foundry.  
Gate IRL Events with Nouns  
Learn how to gate entry to an IRL event for members of a Nounish DAO.  
Hardhat: Analyzing the test coverage of smart contracts  
A tutorial that teaches how to profile the test coverage of your smart contracts using Hardhat and the Solidity Coverage plugin.  
Hardhat: Debugging smart contracts  
A tutorial that teaches how to debug your smart contracts using Hardhat.  
Hardhat: Optimizing the gas usage of smart contracts  
A tutorial that teaches how to optimize the gas usage of your smart contracts using Hardhat.  
Hardhat: Optimizing the size of smart contracts  
A tutorial that teaches how to optimize the size of your smart contracts using Hardhat.  
Thirdweb and Unreal - NFT Items  
Learn how to use NFTs as in-game items using Thirdweb and Unreal.  
Verify a Smart Contract using Basescan API  
A tutorial that teaches how to verify a smart contract using Basescan APIs.  
We use cookies and similar technologies on our websites to enhance and tailor your experience, analyze our traffic, and for security and marketing. You can choose not to allow some type of cookies by clicking Manage Settings. For more information see our Cookie Policy.  
Manage settings  
Accept all

URL: https://docs.base.org/tutorials/tags/smart-wallet  
  
Ecosystem  
Bridge  
Builders  
About  
Socials  
Home  
Get Started  
Docs  
Learn  
Search  
K  
CONNECT  
4 docs tagged with "Smart Wallet"  
View All Tags  
Coinbase Smart Wallet  
Learn to create an app that uses the Coinbase Smart Wallet and effectively manages assets and permissions for both native and new users of onchain apps  
Coinbase Smart Wallet with RainbowKit  
Learn how to configure RainbowKit to elegantly handle the Coinbase Smart Wallet and EOA wallets including the Coinbase wallet, at the same time.  
How to Mint on Zora with an App  
Learn to use Zora contracts inside your app to create secure, efficient, and feature-rich minting experiences for your users.  
Use the Coinbase Smart Wallet and EOAs with OnchainKit  
Learn how to use OnchainKit to easily handle the Coinbase Smart Wallet and EOA wallets including the Coinbase wallet, at the same time.  
We use cookies and similar technologies on our websites to enhance and tailor your experience, analyze our traffic, and for security and marketing. You can choose not to allow some type of cookies by clicking Manage Settings. For more information see our Cookie Policy.  
Manage settings  
Accept all

URL: https://docs.base.org/tutorials/tags/vrf  
  
Ecosystem  
Bridge  
Builders  
About  
Socials  
Home  
Get Started  
Docs  
Learn  
Search  
K  
CONNECT  
One doc tagged with "vrf"  
View All Tags  
Generating random numbers contracts using Supra dVRF  
A tutorial that teaches how to use Supra dVRF to serve random numbers using an onchain randomness generation mechanism directly within your smart contracts on the Base testnet.  
We use cookies and similar technologies on our websites to enhance and tailor your experience, analyze our traffic, and for security and marketing. You can choose not to allow some type of cookies by clicking Manage Settings. For more information see our Cookie Policy.  
Manage settings  
Accept all

URL: https://docs.base.org/base-learn/docs/address-and-payable/  
  
Ecosystem  
Bridge  
Builders  
About  
Socials  
Home  
Get Started  
Docs  
Learn  
Search  
K  
CONNECT  
Get Help on Discord  
Learn to Build Smart Contracts and Onchain Apps  
Introduction to Ethereum  
Development Tools  
Development with Hardhat  
Development With Foundry  
Smart Contract Development  
Overview  
Anatomy of a Smart Contract  
Introduction to Solidity  
Contracts and Basic Functions  
Deploying to a Testnet  
Control Structures  
Storage in Solidity  
Arrays in Solidity  
The Mapping Type  
Advanced Functions  
Structs  
Inheritance  
Imports  
Errors  
The `new` Keyword  
Contract to Contract Interactions  
Events  
Address and Payable  
Address and Payable in Solidity  
Token Development  
Hardhat Tools and Testing  
Onchain App Development (Frontend)  
Exercise Contracts  
Address and Payable in Solidity  
Understanding address and payable address types is crucial for managing Ether transfers and interactions within your Solidity contracts. This article will delve into their key distinctions and practical applications.  
Objectives  
By the end of this lesson, you should be able to:  
Differentiate between address and address payable types in Solidity  
Determine when to use each type appropriately in contract development  
Employ address payable to send Ether and interact with payable functions  
Ethereum Addresses  
In Solidity, Ethereum addresses play a crucial role in interacting with the Ethereum blockchain. An Ethereum address is a 20-byte hexadecimal string that represents the destination of transactions or the owner of a smart contract. These addresses are used to send and receive Ether and interact with smart contracts.  
Addresses  
Regular addresses in Solidity are used for various purposes, including:  
Identifying the owner of a smart contract  
Sending Ether from one address to another  
Checking the balance of an address Here's an example of declaring a regular address variable in Solidity:  
  
address public owner;  
Payable Addresses  
payable keyword is a language-level feature provided by Solidity to enable the handling of Ether within smart contracts, and it is not a feature of the Ethereum Virtual Machine itself, but rather a part of the Solidity language's syntax. They are used when you want a contract to be able to receive Ether from external sources, such as other contracts or user accounts.  
Payable addresses are often used when creating crowdfunding or token sale contracts, where users send Ether to the contract's address in exchange for tokens or to fund a project.  
Here's an example of declaring a payable address variable in Solidity:  
address payable public projectWallet;  
Payable Address are marked as payable, which means they can accept incoming Ether transactions. It's important to note that regular addresses cannot receive Ether directly.  
Receiving Ether with Payable Addresses  
To receive Ether in a contract using a payable address, you need to define a payable function that can accept incoming transactions. This function is typically named receive or fallback. Here's an example:  
fallback() external payable {  
 // Handle the incoming Ether here  
}  
In this example, the fallback function is marked as external and payable, which means it can receive Ether when someone sends it to the contract's address. You can then add custom logic to handle the received Ether, such as updating contract balances or triggering specific actions.  
Usage  
contract PaymentReceiver {  
 address payable owner;  
  
 constructor() payable {  
 owner = payable(msg.sender); // Convert msg.sender to payable  
 }  
  
 function receiveEther() public payable {  
 // This function can receive Ether  
 }  
  
 function withdrawEther() public {  
 owner.transfer(address(this).balance); // Send Ether to owner  
 }  
}  
Conclusion  
Appropriately using address and address payable types is essential for secure and efficient Solidity contract development. By understanding their distinctions and applying them correctly, you can effectively manage Ether transfers and interactions within your contracts.  
ON THIS PAGE  
Ethereum Addresses  
Receiving Ether with Payable Addresses  
Addresses  
Payable Addresses  
Usage  
Conclusion  
Conclusion  
We use cookies and similar technologies on our websites to enhance and tailor your experience, analyze our traffic, and for security and marketing. You can choose not to allow some type of cookies by clicking Manage Settings. For more information see our Cookie Policy.  
Manage settings  
Accept all

URL: https://docs.base.org/base-learn/docs/advanced-functions/function-modifiers  
  
Ecosystem  
Bridge  
Builders  
About  
Socials  
Home  
Get Started  
Docs  
Learn  
Search  
K  
CONNECT  
Get Help on Discord  
Learn to Build Smart Contracts and Onchain Apps  
Introduction to Ethereum  
Development Tools  
Development with Hardhat  
Development With Foundry  
Smart Contract Development  
Overview  
Anatomy of a Smart Contract  
Introduction to Solidity  
Contracts and Basic Functions  
Deploying to a Testnet  
Control Structures  
Storage in Solidity  
Arrays in Solidity  
The Mapping Type  
Advanced Functions  
Function Visibility  
Function Visibility and State Mutability  
Function Modifiers  
Function Modifiers  
Structs  
Inheritance  
Imports  
Errors  
The `new` Keyword  
Contract to Contract Interactions  
Events  
Address and Payable  
Token Development  
Hardhat Tools and Testing  
Onchain App Development (Frontend)  
Exercise Contracts  
Function Modifiers  
Function modifiers allow you to efficiently change the behavior of functions. In some ways, it's similar to inheritance, but there are restrictions, particularly in variable scope.  
Objectives  
By the end of this lesson you should be able to:  
Use modifiers to efficiently add functionality to multiple functions  
Adding a Simple OnlyOwner Modifier  
By default, public functions can be called by anyone, without restriction. Often this is desirable. You want any user to be able to see what NFTs are for sale on your platform, sign up for a service, or read various items stored in state.  
However, there will be many functions you don't want any user to be able to do, such as setting the fee for using the app, or withdrawing all of the funds in the contract! A common pattern to protect these functions is to use modifiers to make sure that only the owner can call these functions.  
CAUTION  
For a production app, you'll want to use a more robust implementation of onlyOwner, such as the one provided by OpenZeppelin.  
Adding an Owner  
The address of the deployer of a contract is not included as an accessible property. To make it available, add it as a state variable and assign msg.sender in the constructor.  
Reveal code  
  
  
  
  
  
  
  
  
Creating an onlyOwner Modifier  
[Modifiers] are very similar to functions and are declared with the modifier keyword. The modifier can run any Solidity code, including functions, and is allowed to modify state. Modifiers must have a special \_ character, which serves as a placeholder for where the code contained within the modified function will run.  
Create a simple onlyOwner modifier, which returns an error of NotOwner with the sending address if the sender is not the owner.  
Reveal code  
  
  
  
  
  
  
  
  
Test your modifier by adding a function that uses it:  
Reveal code  
  
  
  
  
To test, deploy your contract and call the iOwnThis function. You should see the message "You own this!".  
Next, switch the Account, and try the function again. You should see an error in the console:  
call to Modifiers.iOwnThis errored: VM error: revert.  
  
revert  
 The transaction has been reverted to the initial state.  
Error provided by the contract:  
NotOwner  
Parameters:  
{  
 "\_msgSender": {  
 "value": "0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db"  
 }  
}  
Debug the transaction to get more information.  
CAUTION  
Always verify the output of a function call in the console. The result that appears under the button for the function is convenient, but it does not clear or change if a subsequent call reverts.  
Modifiers and Variables  
Modifiers can have parameters, which essentially work the same as in functions. These parameters can be independent values, or they can overlap with the arguments provided to a function call.  
Modifiers with Parameters  
Modifier parameters can be the arguments provided to the functions they modify. You can perform calculations and trigger errors based on these values.  
error NotEven(uint number);  
  
modifier onlyEven(uint \_number) {  
 if(\_number % 2 != 0) {  
 revert NotEven(\_number);  
 }  
 \_;  
}  
  
function halver(uint \_number) public pure onlyEven(\_number) returns (uint) {  
 return \_number / 2;  
}  
Independent Scope  
While modifiers are used to modify functions and can share inputs, they have separate scopes. The following example will not work:  
// Bad code example, does not work  
modifier doubler(uint \_number) {  
 \_number \*= 2;  
 \_;  
}  
  
function modifierDoubler(uint \_number) public pure doubler(\_number) returns (uint) {  
 return \_number; // Returns the original number, NOT number \* 2  
}  
Conclusion  
Function modifiers are an efficient and reusable way to add checks, trigger errors, and control function execution. In this lesson, you've seen examples of how they can be used to abort execution under certain conditions. You've also learned that they have separate scopes and cannot be used to modify variables within the function they modify.  
ON THIS PAGE  
Adding a Simple OnlyOwner Modifier  
Modifiers and Variables  
Modifiers with Parameters  
Independent Scope  
Conclusion  
Modifiers with Parameters  
Independent Scope  
Conclusion  
We use cookies and similar technologies on our websites to enhance and tailor your experience, analyze our traffic, and for security and marketing. You can choose not to allow some type of cookies by clicking Manage Settings. For more information see our Cookie Policy.  
Manage settings  
Accept all

URL: https://docs.base.org/base-learn/docs/advanced-functions/function-modifiers-vid  
  
Ecosystem  
Bridge  
Builders  
About  
Socials  
Home  
Get Started  
Docs  
Learn  
Search  
K  
CONNECT  
Get Help on Discord  
Learn to Build Smart Contracts and Onchain Apps  
Introduction to Ethereum  
Development Tools  
Development with Hardhat  
Development With Foundry  
Smart Contract Development  
Overview  
Anatomy of a Smart Contract  
Introduction to Solidity  
Contracts and Basic Functions  
Deploying to a Testnet  
Control Structures  
Storage in Solidity  
Arrays in Solidity  
The Mapping Type  
Advanced Functions  
Function Visibility  
Function Visibility and State Mutability  
Function Modifiers  
Function Modifiers  
Structs  
Inheritance  
Imports  
Errors  
The `new` Keyword  
Contract to Contract Interactions  
Events  
Address and Payable  
Token Development  
Hardhat Tools and Testing  
Onchain App Development (Frontend)  
Exercise Contracts  
Function Modifiers  
We use cookies and similar technologies on our websites to enhance and tailor your experience, analyze our traffic, and for security and marketing. You can choose not to allow some type of cookies by clicking Manage Settings. For more information see our Cookie Policy.  
Manage settings  
Accept all

URL: https://docs.base.org/base-learn/docs/advanced-functions/function-visibility  
  
Ecosystem  
Bridge  
Builders  
About  
Socials  
Home  
Get Started  
Docs  
Learn  
Search  
K  
CONNECT  
Get Help on Discord  
Learn to Build Smart Contracts and Onchain Apps  
Introduction to Ethereum  
Development Tools  
Development with Hardhat  
Development With Foundry  
Smart Contract Development  
Overview  
Anatomy of a Smart Contract  
Introduction to Solidity  
Contracts and Basic Functions  
Deploying to a Testnet  
Control Structures  
Storage in Solidity  
Arrays in Solidity  
The Mapping Type  
Advanced Functions  
Function Visibility  
Function Visibility and State Mutability  
Function Modifiers  
Function Modifiers  
Structs  
Inheritance  
Imports  
Errors  
The `new` Keyword  
Contract to Contract Interactions  
Events  
Address and Payable  
Token Development  
Hardhat Tools and Testing  
Onchain App Development (Frontend)  
Exercise Contracts  
Function Visibility and State Mutability  
You've seen much of this before, but this document outlines and highlights the options for function visibility and state mutability all in one document.  
Objectives  
By the end of this lesson you should be able to:  
Categorize functions as public, private, internal, or external based on their usage  
Describe how pure and view functions are different than functions that modify storage  
Function Visibility  
There are four types of visibility for functions in Solidity: external, public, internal, and private. These labels represent a further division of the public and private labels you might use in another language.  
External  
Functions with external visibility are only callable from other contracts and cannot be called within their own contract. You may see older references stating you should use external over public because it forces the function to use calldata. This is no longer correct, because both function visibilities can now use calldata and memory for parameters. However, using calldata for either will cost less gas.  
contract Foo {  
 constructor() {  
 // Bad code example, will not work  
 uint bar = foo(3);  
 // ... other code  
 }  
  
 function foo(uint \_number) external pure returns (uint) {  
 return \_number\*2;  
 }  
}  
Public  
Public functions work the same as external, except they may also be called within the contract that contains them.  
contract Foo {  
 constructor() {  
 // Public functions may be called within the contract  
 uint bar = foo(3);  
 // ... other code  
 }  
  
 function foo(uint \_number) public pure returns (uint) {  
 return \_number\*2;  
 }  
}  
Private and Internal  
Functions visible as private and internal operate nearly identically. Beyond writing hygienic code, these have a very important effect. Because they are not a part of the contract's ABI, you can use mappings and storage variable references as parameters.  
The difference is that private functions can't be called from derived contracts. You'll learn more about that when we cover inheritance.  
Some developers prepend an underscore to private and internal functions.  
function \_foo(uint \_number) private returns (uint) {  
 return \_number\*2;  
}  
DANGER  
All data on a blockchain is public. Don't mistake hiding visibility while coding for hiding information from the world!  
Function State Mutability  
State mutability labels are relatively unique to Solidity. They determine how a function can interact with state, which has a substantial impact on gas costs.  
Pure  
pure functions promise to neither read nor write state. They're usually used for helper functions that support other functionality.  
function abs(int x) public pure returns (int) {  
 return x >= 0 ? x : -x;  
}  
pure functions can be called from outside the blockchain without using gas, if they are also public or external.  
View  
view functions access state, but don't modify it. You've used these for tasks such as returning all the values in an array.  
function getArr() public view returns (uint[] memory) {  
 return arr;  
}  
view functions can be called from outside the blockchain without using gas, if they are also public or external.  
Unlabeled Functions  
Functions that are not labeled view or pure can modify state and the compiler will generate a warning if they do not.  
function addToArr(uint \_number) public {  
 arr.push(\_number);  
}  
They can have any visibility and will always cost gas when called.  
Conclusion  
The visibility and mutability keywords in Solidity help you organize your code and alert other developers to the properties of each of your functions. Use them to keep your code organized and readable.  
ON THIS PAGE  
Function Visibility  
Function State Mutability  
Pure  
View  
Unlabeled Functions  
Conclusion  
Pure  
View  
Unlabeled Functions  
Conclusion  
We use cookies and similar technologies on our websites to enhance and tailor your experience, analyze our traffic, and for security and marketing. You can choose not to allow some type of cookies by clicking Manage Settings. For more information see our Cookie Policy.  
Manage settings  
Accept all

URL: https://docs.base.org/base-learn/docs/advanced-functions/function-visibility-vid  
  
Ecosystem  
Bridge  
Builders  
About  
Socials  
Home  
Get Started  
Docs  
Learn  
Search  
K  
CONNECT  
Get Help on Discord  
Learn to Build Smart Contracts and Onchain Apps  
Introduction to Ethereum  
Development Tools  
Development with Hardhat  
Development With Foundry  
Smart Contract Development  
Overview  
Anatomy of a Smart Contract  
Introduction to Solidity  
Contracts and Basic Functions  
Deploying to a Testnet  
Control Structures  
Storage in Solidity  
Arrays in Solidity  
The Mapping Type  
Advanced Functions  
Function Visibility  
Function Visibility and State Mutability  
Function Modifiers  
Function Modifiers  
Structs  
Inheritance  
Imports  
Errors  
The `new` Keyword  
Contract to Contract Interactions  
Events  
Address and Payable  
Token Development  
Hardhat Tools and Testing  
Onchain App Development (Frontend)  
Exercise Contracts  
Function Visibility  
We use cookies and similar technologies on our websites to enhance and tailor your experience, analyze our traffic, and for security and marketing. You can choose not to allow some type of cookies by clicking Manage Settings. For more information see our Cookie Policy.  
Manage settings  
Accept all

URL: https://docs.base.org/base-learn/docs/arrays/array-storage-layout-vid  
  
Ecosystem  
Bridge  
Builders  
About  
Socials  
Home  
Get Started  
Docs  
Learn  
Search  
K  
CONNECT  
Get Help on Discord  
Learn to Build Smart Contracts and Onchain Apps  
Introduction to Ethereum  
Development Tools  
Development with Hardhat  
Development With Foundry  
Smart Contract Development  
Overview  
Anatomy of a Smart Contract  
Introduction to Solidity  
Contracts and Basic Functions  
Deploying to a Testnet  
Control Structures  
Storage in Solidity  
Arrays in Solidity  
Arrays  
Writing Arrays  
Arrays  
Filtering an Array  
Fixed-Size Arrays  
Array Storage Layout  
Arrays Exercise  
The Mapping Type  
Advanced Functions  
Structs  
Inheritance  
Imports  
Errors  
The `new` Keyword  
Contract to Contract Interactions  
Events  
Address and Payable  
Token Development  
Hardhat Tools and Testing  
Onchain App Development (Frontend)  
Exercise Contracts  
Array Storage Layout  
We use cookies and similar technologies on our websites to enhance and tailor your experience, analyze our traffic, and for security and marketing. You can choose not to allow some type of cookies by clicking Manage Settings. For more information see our Cookie Policy.  
Manage settings  
Accept all

URL: https://docs.base.org/base-learn/docs/arrays/arrays-exercise  
  
Ecosystem  
Bridge  
Builders  
About  
Socials  
Home  
Get Started  
Docs  
Learn  
Search  
K  
CONNECT  
Get Help on Discord  
Learn to Build Smart Contracts and Onchain Apps  
Introduction to Ethereum  
Development Tools  
Development with Hardhat  
Development With Foundry  
Smart Contract Development  
Overview  
Anatomy of a Smart Contract  
Introduction to Solidity  
Contracts and Basic Functions  
Deploying to a Testnet  
Control Structures  
Storage in Solidity  
Arrays in Solidity  
Arrays  
Writing Arrays  
Arrays  
Filtering an Array  
Fixed-Size Arrays  
Array Storage Layout  
Arrays Exercise  
The Mapping Type  
Advanced Functions  
Structs  
Inheritance  
Imports  
Errors  
The `new` Keyword  
Contract to Contract Interactions  
Events  
Address and Payable  
Token Development  
Hardhat Tools and Testing  
Onchain App Development (Frontend)  
Exercise Contracts  
Arrays Exercise  
Create a contract that adheres to the following specifications.  
Contract  
Review the contract in the starter snippet called ArraysExercise. It contains an array called numbers that is initialized with the numbers 1–10. Copy and paste this into your file.  
contract ArraysExercise {  
 uint[] public numbers = [1,2,3,4,5,6,7,8,9,10];  
}  
Add the following functions:  
Return a Complete Array  
The compiler automatically adds a getter for individual elements in the array, but it does not automatically provide functionality to retrieve the entire array.  
Write a function called getNumbers that returns the entire numbers array.  
Reset Numbers  
Write a public function called resetNumbers that resets the numbers array to its initial value, holding the numbers from 1-10.  
NOTE  
We'll award the pin for any solution that works, but one that doesn't use .push() is more gas-efficient!  
CAUTION  
Remember, anyone can call a public function! You'll learn how to protect functionality in another lesson.  
Append to an Existing Array  
Write a function called appendToNumbers that takes a uint[] calldata array called \_toAppend, and adds that array to the storage array called numbers, already present in the starter.  
Timestamp Saving  
At the contract level, add an address array called senders and a uint array called timestamps.  
Write a function called saveTimestamp that takes a uint called \_unixTimestamp as an argument. When called, it should add the address of the caller to the end of senders and the \_unixTimeStamp to timestamps.  
TIP  
You'll need to research on your own to discover the correct Special Variables and Functions that can help you with this challenge!  
Timestamp Filtering  
Write a function called afterY2K that takes no arguments. When called, it should return two arrays.  
The first should return all timestamps that are more recent than January 1, 2000, 12:00am. To save you a click, the Unix timestamp for this date and time is 946702800.  
The second should return a list of senders addresses corresponding to those timestamps.  
Resets  
Add public functions called resetSenders and resetTimestamps that reset those storage variables.  
Submit your Contract and Earn an NFT Badge! (BETA)  
INFO  
Hey, where'd my NFT go!?  
Testnets are not permanent! Base Goerli will soon be sunset, in favor of Base Sepolia.  
As these are separate networks with separate data, your NFTs will not transfer over.  
Don't worry! We've captured the addresses of all NFT owners on Base Goerli and will include them when we release the mechanism to transfer these NFTs to mainnet later this year! You can also redeploy on Sepolia and resubmit if you'd like!  
Connect Wallet  
Please connect your wallet.  
If you need a wallet, you can get the Coinbase Wallet here.  
You can also use MetaMask and other popular wallets.  
ON THIS PAGE  
Contract  
Return a Complete Array  
Reset Numbers  
Append to an Existing Array  
Timestamp Saving  
Timestamp Filtering  
Resets  
Submit your Contract and Earn an NFT Badge! (BETA)  
We use cookies and similar technologies on our websites to enhance and tailor your experience, analyze our traffic, and for security and marketing. You can choose not to allow some type of cookies by clicking Manage Settings. For more information see our Cookie Policy.  
Manage settings  
Accept all

URL: https://docs.base.org/base-learn/docs/arrays/arrays-in-solidity  
  
Ecosystem  
Bridge  
Builders  
About  
Socials  
Home  
Get Started  
Docs  
Learn  
Search  
K  
CONNECT  
Get Help on Discord  
Learn to Build Smart Contracts and Onchain Apps  
Introduction to Ethereum  
Development Tools  
Development with Hardhat  
Development With Foundry  
Smart Contract Development  
Overview  
Anatomy of a Smart Contract  
Introduction to Solidity  
Contracts and Basic Functions  
Deploying to a Testnet  
Control Structures  
Storage in Solidity  
Arrays in Solidity  
Arrays  
Writing Arrays  
Arrays  
Filtering an Array  
Fixed-Size Arrays  
Array Storage Layout  
Arrays Exercise  
The Mapping Type  
Advanced Functions  
Structs  
Inheritance  
Imports  
Errors  
The `new` Keyword  
Contract to Contract Interactions  
Events  
Address and Payable  
Token Development  
Hardhat Tools and Testing  
Onchain App Development (Frontend)  
Exercise Contracts  
Arrays  
Solidity arrays are collections of the same type, accessed via an index, the same as any other language. Unlike other languages, there are three types of arrays - storage, memory, and calldata. Each has their own properties and constraints.  
Objectives  
By the end of this lesson you should be able to:  
Describe the difference between storage, memory, and calldata arrays  
Storage, Memory, and Calldata  
The storage, memory, or calldata keywords are required when declaring a new reference type variable. This keyword determines the data location where the variable is stored and how long it will persist.  
Storage  
The storage keyword is used to assign state variables that become a part of the blockchain as a part of the storage for your contract. These remain as assigned until modified, for the lifetime of the contract.  
Storage is very expensive compared to most other environments. It costs a minimum of 20000 gas to store a value in a new storage slot, though it's cheaper to update that value after the initial assignment (~5000+ gas).  
This cost isn't a reason to be afraid of using storage. In the long run, writing clear, maintainable, and logical code will always cost less than jumping through hoops to save gas here and there. Just be as thoughtful with storage on the EVM as you would be with computation in most other environments.  
Memory  
The memory keyword creates temporary variables that only exist within the scope in which they are created. Memory is less expensive than storage, although this is relative. There are often circumstances where it is cheaper to work directly in storage rather than convert to memory and back. Copying from one location to another can be quite expensive!  
Calldata  
The calldata storage location is where function arguments are stored. It is non-modifiable and the Solidity docs recommend using it where possible to avoid unnecessary copying, because it can't be modified. You'll learn more about this later, but doing so can help prevent confusing bugs when calling a function from another contract that takes in values from that contract's storage.  
Array Data Locations  
Arrays behave differently based on their data location. Assignment behavior also depends on data location. To quote the docs:  
Assignments between storage and memory (or from calldata) always create an independent copy.  
Assignments from memory to memory only create references. This means that changes to one memory variable are also visible in all other memory variables that refer to the same data.  
Assignments from storage to a local storage variable also only assign a reference.  
All other assignments to storage always copy. Examples for this case are assignments to state variables or to members of local variables of storage struct type, even if the local variable itself is just a reference.  
Storage Arrays  
Arrays in storage are passed by reference. In other words, if you assign a storage array half a dozen names, any changes you make will always modify the original, underlying storage array.  
contract StorageArray {  
 // Variables declared at the class level are always `storage`  
 uint[] arr = [1, 2, 3, 4, 5];  
  
 function function\_1() public {  
 uint[] storage arr2 = arr;  
  
 arr2[0] = 99; // <- arr is now [99, 2, 3, 4, 5];  
 }  
}  
You cannot use a storage array as a function parameter, and you cannot write a function that returns a storage array.  
Storage arrays are dynamic, unless they are declared with an explicit size. However, their functionality is limited compared to other languages. The .push(value) function works as expected. the .pop() function removes the last value of an array, but it does not return that value. You also may not use .pop() with an index to remove an element from the middle of an array, or to remove more than one element.  
You can use the delete keyword with an array. Doing so on an entire array will reset the array to zero length. Calling it on an element within the array will reset that value to its default. It will not resize the array!  
uint[] arr\_2 = [1, 2, 3, 4, 5];  
function function\_2(uint \_num) public returns(uint[] memory) {  
 arr\_2.push(\_num); // <- arr\_2 is [1, 2, 3, 4, 5, <\_num>]  
  
 delete arr\_2[2]; // <- arr\_2 is [1, 2, 0, 4, 5, <\_num>]  
  
 arr\_2.pop(); // <- arr\_2 is [1, 2, 0, 4, 5] (\_num IS NOT returned by .pop())  
  
 delete arr\_2; // <- arr\_2 is []  
  
 return arr\_2; // <- returns []  
}  
Storage arrays are implicitly convertible to memory arrays.  
Memory Arrays  
Arrays declared as memory are temporary and only exist within the scope in which they are created. Arrays in memory are not dynamic and must be declared with a fixed size. This can be done at compile time, by declaring a size inside the [] or during runtime by using the new keyword. Finally, memory arrays can be implicitly cast from storage arrays.  
function function\_3(uint \_arrSize) public {  
 uint[5] memory arrSizeFive;  
 uint[] memory arrWithCustomSize = new uint[](\_arrSize);  
 uint[] memory localCopyOfArr = arr;  
 // ...do something  
}  
The declaration pattern impacts gas cost, though keep in mind that the first two examples are empty, and would cost additional gas depending on how they are eventually filled.  
function declareMemoryArrays() public view {  
 uint[5] memory simpleArr; // this line costs 135 gas  
 uint[] memory emptyArr = new uint[](5); // This line costs 194 gas  
 uint[] memory arrCopy = arr; // This line costs 13166 gas  
}  
The lack of dynamic memory arrays can require some gymnastics if you need to create an array where the size is not initially known. Depending on the specific needs of the problem, valid solutions for filtering an array and returning a smaller array could include:  
Looping through a larger array twice, first to count the number, then to copy the appropriate elements  
Tracking the number of elements that meet condition X with a storage variable, then instantiating the array with <type>[] memory filteredArray = new <type>[](numX);  
Using multiple data structures to track references to different subsets  
Calldata Arrays  
Arrays in calldata are read only. Otherwise, they function the same as any other array.  
Array slices are currently only implemented for calldata arrays.  
Conclusion  
In this lesson, you've learned the differences between the memory, storage, and calldata data locations. You've also learned how they apply to arrays, with each having its own properties, restrictions, and costs.  
ON THIS PAGE  
Storage, Memory, and Calldata  
Array Data Locations  
Storage Arrays  
Memory Arrays  
Calldata Arrays  
Conclusion  
Storage Arrays  
Memory Arrays  
Calldata Arrays  
Conclusion  
We use cookies and similar technologies on our websites to enhance and tailor your experience, analyze our traffic, and for security and marketing. You can choose not to allow some type of cookies by clicking Manage Settings. For more information see our Cookie Policy.  
Manage settings  
Accept all

URL: https://docs.base.org/base-learn/docs/arrays/arrays-in-solidity-vid  
  
Ecosystem  
Bridge  
Builders  
About  
Socials  
Home  
Get Started  
Docs  
Learn  
Search  
K  
CONNECT  
Get Help on Discord  
Learn to Build Smart Contracts and Onchain Apps  
Introduction to Ethereum  
Development Tools  
Development with Hardhat  
Development With Foundry  
Smart Contract Development  
Overview  
Anatomy of a Smart Contract  
Introduction to Solidity  
Contracts and Basic Functions  
Deploying to a Testnet  
Control Structures  
Storage in Solidity  
Arrays in Solidity  
Arrays  
Writing Arrays  
Arrays  
Filtering an Array  
Fixed-Size Arrays  
Array Storage Layout  
Arrays Exercise  
The Mapping Type  
Advanced Functions  
Structs  
Inheritance  
Imports  
Errors  
The `new` Keyword  
Contract to Contract Interactions  
Events  
Address and Payable  
Token Development  
Hardhat Tools and Testing  
Onchain App Development (Frontend)  
Exercise Contracts  
Arrays  
We use cookies and similar technologies on our websites to enhance and tailor your experience, analyze our traffic, and for security and marketing. You can choose not to allow some type of cookies by clicking Manage Settings. For more information see our Cookie Policy.  
Manage settings  
Accept all

URL: https://docs.base.org/base-learn/docs/arrays/filtering-an-array-sbs  
  
Ecosystem  
Bridge  
Builders  
About  
Socials  
Home  
Get Started  
Docs  
Learn  
Search  
K  
CONNECT  
Get Help on Discord  
Learn to Build Smart Contracts and Onchain Apps  
Introduction to Ethereum  
Development Tools  
Development with Hardhat  
Development With Foundry  
Smart Contract Development  
Overview  
Anatomy of a Smart Contract  
Introduction to Solidity  
Contracts and Basic Functions  
Deploying to a Testnet  
Control Structures  
Storage in Solidity  
Arrays in Solidity  
Arrays  
Writing Arrays  
Arrays  
Filtering an Array  
Fixed-Size Arrays  
Array Storage Layout  
Arrays Exercise  
The Mapping Type  
Advanced Functions  
Structs  
Inheritance  
Imports  
Errors  
The `new` Keyword  
Contract to Contract Interactions  
Events  
Address and Payable  
Token Development  
Hardhat Tools and Testing  
Onchain App Development (Frontend)  
Exercise Contracts  
Filtering an Array  
In this exercise, you'll explore two different solutions for filtering an array in Solidity. By doing so, you'll gain a better understanding of the constraints present while working with arrays, and have the chance to learn and compare the gas costs of different approaches.  
Objectives  
By the end of this lesson you should be able to:  
Write a function that can return a filtered subset of an array  
First Pass Solution  
Setup  
Create a new workspace in Remix and add a file called ArrayDemo.sol containing a contract called ArrayDemo. Initialize an array containing the numbers from 1 to 10. Add a stub for a function called getEvenNumbers that returns a uint[] memory.  
contract ArrayDemo {  
 uint[] public numbers = [1,2,3,4,5,6,7,8,9,10];  
  
 function getEvenNumbers() external view returns(uint[] memory) {  
 // TODO  
 }  
}  
CAUTION  
You don't have to declare the size of the memory array to be returned. You usually don't want to either, unless the results will always be the same, known size.  
Finding the Number of Even Numbers  
We need to initialize a memory array to hold the results, but to do so, we need to know how big to make the array. Don't be tempted to count the number of evens in numbers, as what happens if we modify it later?  
The simple and obvious solution is to simply iterate through numbers and count how many even numbers are present. You could add that functionality in getEvenNumbers(), but it might be useful elsewhere, so a better practice would be to separate these concerns into another function.  
Go ahead and write it on your own. It needs to:  
Instantiate a uint to hold the results  
Iterate through all of the values in numbers and increment that number if the value is even  
Return the result  
You should end up with something like:  
Reveal code  
  
  
  
  
  
  
  
  
  
  
  
  
The \_ in front of the function name is a practice used by some developers, in Solidity and in other languages, to indicate visually that this function is intended for internal use only.  
Returning Only Even Numbers  
Now that we have a method to find out how big the return array needs to be in getEvenNumbers(), we can simply loop through numbers, and add the even numbers to the array to be returned.  
Finish the function on your own. It needs to:  
Determine the number of results and instantiate an array that size  
Loop through the numbers array and if a given number is even, add it to the next unused index in the results array  
You should end up with something like:  
Reveal code  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
Did you catch the compiler warning about view? You aren't modifying state, so you should mark it as such.  
Testing the Function  
Deploy your contract and test the function. You should get a return of [2,4,6,8,10]. The total gas cost will be about 63,947, depending on if you used the same helper variables, etc.  
Optimizing the Function  
It does seem inefficient to loop through the same array twice. What if we instead kept track of how many even numbers to expect. That way, we would only need to loop once, thus saving gas! Right?  
Only one way to find out.  
Tracking Relevant Data  
Add a contract-level variable called numEven, and initialize it with 5, the number of even numbers in the array. Modify getEvenNumbers() to use numEven instead of the \_countEvenNumbers() function. It should now look like:  
Reveal code  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
Redeploy and test again. Success, the function now only costs about 57,484 gas to run! Except there is a catch. Remember, it's going to cost about 5000 gas to update numEven each time the array adds an even number.  
A More Realistic Accounting  
As we considered above, in a real-world example, we wouldn't declare the array up front, it would be modified over time. A slightly more realistic example would be to fill the array with a function.  
Change the declaration for numbers and numEven so that they have their respective default values to begin with.  
uint[] public numbers;  
uint numEven;  
Add a new function called debugLoadArray that takes a uint called \_number as an argument, and fills the array by looping through \_numbers times, pushing each number into the array. For now, don't update numEven.  
Reveal code  
  
  
  
  
  
  
Test out the function by loading in 10 numbers. It costs about 249,610 gas to load the array. Now, add functionality to also increment numEven when the number added is even. We can't just calculate it, because although the numbers are sequential in the debug function, they might not be in real world use.  
Reveal code  
  
  
  
  
  
  
  
  
  
Be sure to redeploy and try again with 10 numbers. This time, the cost was about 275,335 gas. That's almost 26,000 more gas in an effort to save the 5,000 gas needed to run \_countEvenNumbers().  
Looking at the Big Picture  
What about more? What if there are a thousand numbers in the array? What about a million?  
Let's start with 500, any more will break the Remix EVM simulation, and/or would trigger an out of gas error because we're approaching the gas limit for the entire block.  
Comment out the if statement in debugLoadArray that checks for even numbers and load 500 numbers. The Remix EVM should be able to handle this, but it might hang up for a moment, or even crash. (You can also do this experiment with 250 numbers instead.)  
function debugLoadArray(uint \_number) external {  
 for(uint i = 0; i < \_number; i++) {  
 numbers.push(i);  
 // if(i % 2 == 0) {  
 // numEven++;  
 //}  
 }  
}  
You'll get a result of about 11,323,132 gas to load the array. That's a lot! The target total gas for a single block is 15 million, and the limit is 30 million.  
Try again with the code to increment numEven. You should get about 11,536,282, or an increase of about 213,150 gas.  
Now, test out getEvenNumbers() using numEven vs. using \_countEvenNumbers(). With numEven, it should cost about 1,578,741 gas to find the even numbers. Using \_countEvenNumbers(), that cost increases to 1,995,579 gas, an increase of 416,838 gas.  
Which is Better?  
As is often the case with code, it depends. You might think that the experiment makes things obvious. Paying 213k gas up front to track \_numEven results in a savings of over 400k gas when filtering for even numbers. Even better, you might realize that the upfront cost difference will be spread across all of your users over time, making them almost trivial. You also might think that it's possible that the filter function could be called dozens of times for each time 500 numbers are loaded.  
These are all valid considerations that you should evaluate as you are developing your code solution to a business problem. One last critical element to consider is that there is only a gas cost to read from the blockchain if it's another contract calling the function. It doesn't cost any gas to call view or pure functions from a front end or app.  
If getEvenNumbers will never be called by another contract, then using numEven might cost more for no benefit!  
Conclusion  
In this lesson, you've explored a few different approaches to a problem. You've learned how to filter an array, but more importantly, you've learned some of the specific considerations in blockchain development. Finally, you've seen that pushing 500 integers to an array, usually a trivial operation, is very large and very expensive on the EVM.  
ON THIS PAGE  
First Pass Solution  
Optimizing the Function  
Tracking Relevant Data  
A More Realistic Accounting  
Looking at the Big Picture  
Which is Better?  
Conclusion  
Tracking Relevant Data  
A More Realistic Accounting  
Looking at the Big Picture  
Which is Better?  
Conclusion  
We use cookies and similar technologies on our websites to enhance and tailor your experience, analyze our traffic, and for security and marketing. You can choose not to allow some type of cook